Resumo das seções 2, 3 e 4 do curso de Início Rápido em Teste e QA

- Todo mundo deve ajudar a testar o sistema a fim de promover uma maior qualidade, entretanto, um QA possui habilidades diferentes de investigar e detectar defeitos - tomando ações para prevenir que os erros aconteçam, visto que já identificam os padrões. Assim sendo, os outros profissionais utilizam o sistema e notam defeitos por meio da análise se o sistema está funcionando corretamente, enquanto o profissional QA busca ativamente os defeitos, analisando o que pode dar errado.
- É necessário conhecer o negócio que você testa, então é importante que o testador estude e conheça o funcionamento do ambiente do sistema a ser testado. Dessa forma, é relevante conhecer normas, modelos de trabalho, avaliações e certificações que fazem parte do negócio, a fim de entender o que a empresa deve seguir e o que pode implicar na sua penalização. Além disso, ter conhecimento a respeito de sistemas operacionais, ferramentas Office, idiomas (principalmente inglês, espanhol e mandarim - diferencial), lógica e linguagem de programação, infraestrutura.
- Vivenciar o ambiente de uso real do software por, pelo menos, 1 dia, pode proporcionar uma visão ampla dos problemas enfrentados e de como o sistema pode, de fato, auxiliar de maneira adequada na realização de tarefas.
- Uma pessoa da área da tecnologia deve estar constantemente estudando, buscando novas ferramentas e tecnologias emergentes.
- Um QA deve ter soft skills ligadas, principalmente, à comunicação e trabalho em equipe, uma vez que suas atividades se baseiam no planejamento, análise, modelagem e execução, relatando erros e inconsistências por meio da escrita e comunicação com a equipe. É importante ressaltar a relevância de características como motivação, resistência, capacidade de aprender rapidamente, curiosidade, detalhismo e perfeccionismo, as quais permitem

que o profissional confie no seu próprio potencial e consiga aplicar suas habilidades de maneira minuciosa e pontual.

- **Débito Técnico:** falta de habilidades e conhecimento de ferramentas para exercer uma profissão.

A aparição de bugs no sistema pode impactar negativamente diversas áreas, causando danos como:

- atrasos nas entregas de software de qualidade funcionando, o que gera uma perda de confiança das empresas/organizações que necessitam de um sistema adequado.
- prejuízo financeiro, visto que são gerados muitos custos em correção de bugs e novos testes
- prejuízo de imagem, dado que o responsável pela tarefa acaba sendo associado a entregas de baixa qualidade.
- invasão de pessoas não autorizadas.
- vazamento de informações sigilosas.
- em caso de Governos, decisões estratégicas incorretas.
- risco de vida e acidentes, uma vez que sistemas para auxílio de atividades de alto risco ou de alta confidencialidade devem ser, obrigatoriamente, confiáveis. Para exemplificar, pode-se citar um sistema para controle de aviões: se o software mostrar a rota de um determinado avião de modo incorreto, o risco de um acidente grave ocorrer é iminente, já que o avião em questão pode acabar entrando na rota de outro e causando uma colisão.
- desperdício de recursos e poluição, tendo em vista softwares que não desligam, softwares que calculam errado a quantia de recursos necessários e acabam gastando mais do que o devido ao produzir um determinado produto, etc.

ISTQB - entidade com sede na Bélgica que criou sete fundamentos do teste, caracterizados por conceitos e ideias que estabelecem o que o teste consegue ou não fazer:

- **Teste demonstra a presença de defeitos, mas nunca a sua ausência:** o teste pode demonstrar a presença de defeitos, mas não garante que eles não existem, o que significa que podem ser realizados diversos testes com conhecimentos de alto nível aplicados e, mesmo assim, pode ser que alguns defeitos ainda passem despercebidos. Assim sendo, o profissional QA deve ter sempre uma desconfiança de que existem outros bugs, uma vez que sua meta é promover um sistema perfeito. Em casos de sistemas de alto risco, os

testes devem ser exaustivos, pois o software precisa funcionar perfeitamente para evitar erros, Já em casos de softwares comerciais, o limite de testes deve ser alinhado com o limite de custos da empresa.

- **Teste exaustivo não é possível:** testar tudo não é viável, exceto para casos triviais. Tendo isso em vista, é necessário levar em conta riscos e prioridades - com base, principalmente, no que se usa mais e o que é mais importante para a empresa - para dar foco aos esforços de teste.
- **Teste antecipado:** quanto mais breve a atividade de teste começar, mais retorno vai ser recebido. Se um bug não for reportado e corrigido logo no início do processo, este vai acarretar em outros, o que vai resultar em gastos que poderiam ser evitados anteriormente.
- **Agrupamento de defeitos:** os bugs não são distribuídos de maneira homogênea pelo sistema, o que significa que alguns módulos possuem mais defeitos que outros. Assim sendo, é interessante procurar defeitos em lugares que já apresentaram bugs, lugares com maior índice de reclamações de clientes e em locais de alto risco - os quais podem gerar grandes consequências e prejuízos.
- **Paradoxo do pesticida:** realizar os mesmos testes sempre faz com que eles percam a eficácia, o adequado é atualizar os casos de teste conforme o software evolui.
- **Teste depende do contexto:** os testes são realizados de acordo com o seu contexto, ou seja, quanto maior o risco, mais testes.
- **A ilusão da ausência de erros:** mesmo que os bugs sejam encontrados e consertados, é importante que o sistema atenda, efetivamente, às expectativas e necessidades do usuário.

Erro: engano cometido por uma pessoa que só pode ser reconhecido por quem o cometeu.

Defeito: problema encontrado no trabalho de outra pessoa.

Falha: quando um defeito é executado e causa um problema visível no sistema.

IEC/ISO 25010:

Verificadas em Testes Funcionais:

- **Adequação funcional (anteriormente chamada de Funcionalidade):** característica referente a uma funcionalidade adequada ao que foi pedido, ou seja, aquela que cumpre o seu propósito. Essas características representam uma parte de negócio, não um aspecto técnico.
    - Analisa a **completude**, **correção** e **apropriação** - se as informações são exibidas de maneira adequada - as funcionalidades de um software.

Verificadas em Testes Não-Funcionais:

- **Usabilidade:** característica referente à facilidade com que o usuário tem de utilizar o sistema, sem que este necessite de manual.
    - **Reconhecibilidade:** facilidade do usuário ao reconhecer elementos da tela e comportamentos.
    - **Aprendizibilidade:** capacidade do software de ensinar o usuário - como ícones de interrogação que informam para que um determinado campo serve.
    - **Operabilidade:** facilidade de operação e navegação no sistema - como menos cliques para realizar uma operação.
    - **Proteção contra erro do usuário:** sistemas com meios de não permitir o usuário cometa determinados erros - como uma lista para selecionar estados, evitando que a pessoa insira um nome incorreto.
    - **Estética (da interface do usuário)**
    - **Acessibilidade:** facilidade de acesso de todas as pessoas ao software.
- **Compatibilidade:** característica referente à capacidade de um software ser compatível com outros softwares.
    - **Coexistência:** facilidade de coexistir com outros sistemas.
    - **Interoperabilidade:** facilidade de comunicação entre os sistemas.
- **Confiança ou Confiabilidade:** característica referente a um software que está sempre disponível para o usuário.
    - **Maturidade:** capacidade de perceber e prevenir a falha antes que ela aconteça - aviso de que uma coisa vai dar errado.
    - **Disponibilidade:** capacidade do sistema se manter à disposição de usuários e sistemas.
    - **Tolerância a falhas:** capacidade de perceber e compensar as falhas em tempo real.
    - **Recuperabilidade:** capacidade de se recuperar de falhas e travamentos.
- **Eficiência (de desempenho):** característica referente à capacidade de um software funcionar rapidamente.
    - **Comportamento em relação ao tempo:** performance em si do software.
    - **Utilização de recursos:** observar como o usuário utiliza os recursos disponíveis - é melhor um software que usa muitos recursos do que um software que possui recursos mas não os utiliza.
    - **Capacidade:** capacidade do software de atender a transações e usuários.
- **Manutenibilidade:** característica referente à facilidade de submeter um software à manutenção.
    - **Modularidade:** software dividido em módulos/partes.

    - **Reusabilidade:** facilidade do software em ser reutilizado em outros lugares.
    - **Analisabilidade:** facilidade de analisar o programa, se o código é fácil de compreender.
    - **Modificabilidade:** facilidade em modificar o software, se é possível mudar os componentes de maneira simples.
    - **Testabilidade:** facilidade de testar um software - questionar a testabilidade na hora que receber a documentação para evitar futuros problemas.
- **Portabilidade:** característica referente à capacidade de um software de funcionar em vários sistemas operacionais, navegadores e dispositivos.
    - **Adaptabilidade:** facilidade do software funcionar em um ambiente novo, de se adaptar.
    - **Instalabilidade:** facilidade de instalar ou desinstalar uma aplicação.
    - **Substituibilidade:** facilidade de substituir um software por um novo.
- **Segurança:** característica referente à capacidade de um software não ser invadido e/ou manipulado.
    - **Confidencialidade:** capacidade de que apenas quem criou ou quem tem uma hierarquia maior tenha acesso a uma determinada informação.
    - **Integridade:** capacidade de que apenas pessoas autorizadas podem modificar determinadas informações e, se houver alguma mudança, esta deve ser registrada.
    - **Não repúdio:** garantia que a pessoa que está fazendo uma transação é realmente um usuário específico do sistema.
    - **Responsabilidade:** garantir que uma pessoa que utilizou o software fez uma determinada ação de uma forma - por meio de um log, por exemplo.
    - **Autenticidade:** garantia das ações realizadas no sistema.

Comprometido x Envolvido: Comprometido é aquele que se interessa pela sua função, que está sempre buscando aprimorar os seus conhecimentos e impactar positivamente o seu ambiente de trabalho. Já o envolvido é aquele que apenas vai conforte o fluxo, não se esforça para realizar as suas atividades e tem um pensamento individualista.
Autogerenciamento: gestão de tempo, energia, recursos, tarefas, restrições e compromissos para promover uma maior organização pessoal e, consequentemente, profissional.
Negociação: buscar um meio de que o resultado se torne favorável, mesmo que parcialmente, para todas as partes.
Produtividade: Priorizar e dividir seu tempo para a realização das atividades diárias, sabendo quando realizar uma menor quantidade de atividades para manter a produtividade.